

**МИНОБРНАУКИ РОССИИ**  
**Ярославский государственный университет им. П.Г. Демидова**

Кафедра компьютерной безопасности и математических методов обработки информации

УТВЕРЖДАЮ

Декан математического факультета

\_\_\_\_\_  
Нестеров П.Н.

23 мая 2023 г.

**Рабочая программа дисциплины**

**Языки программирования**

Направление подготовки (специальности)  
10.03.01 Информационная безопасность

Направленность (профиль)  
«Безопасность компьютерных систем (в сфере информационных технологий)»

Форма обучения очная

Программа рассмотрена  
на заседании кафедры  
от 14.04.2023, протокол № 8

Программа одобрена НМК  
математического факультета  
протокол № 9 от 03.05.2023

## 1. Цели освоения дисциплины

*Цель дисциплины.* Дисциплина «Языки программирования» нацелена на подготовку специалистов к деятельности, связанной с разработкой программного обеспечения для решения профессиональных задач.

Курс должен дать сформировать фундаментальную базу, необходимую для успешного освоения дисциплин, изучение которых связано с созданием информационных систем для различных предметных областей, их анализом, внедрением и сопровождением.

*Задачи дисциплины:*

- изучение общих принципов построения и использования современных языков программирования;
- изучение основ алгоритмизации;
- изучение средств описания данных и средств описания действий языков программирования высокого уровня;
- изучение объектно-ориентированного программирования;
- изучение языка ассемблера персонального компьютера;
- овладение навыками программирования;
- освоение современных сред создания программных продуктов.

*Целью воспитания личности* при реализации программы дисциплины является формирование таких черт как организованность и умение планировать время для выполнения сложных проектов; умение общаться с людьми в ходе выполнения этапа анализа предметной области и при подготовке рекомендаций по использованию созданных приложений, трудолюбие, ответственность, способность к саморазвитию, повышению своей квалификации и мастерства.

## 2. Место дисциплины в структуре образовательной программы

Дисциплина «Языки программирования» относится к обязательной части образовательной программы.

Для успешного усвоения данной дисциплины необходимо, чтобы студент владел знаниями, умениями и навыками, сформированными в процессе изучения дисциплин:

«Математический анализ», «Алгебра», «Геометрия» - знать основные понятия и методы математического анализа, алгебры, геометрии; уметь решать задачи теории пределов функций, дифференцирования, интегрирования и разложения функций в ряды; владеть навыками использования стандартных методов и моделей математического анализа и их применения к решению прикладных задач;

«Информатика» - знать состав, назначение функциональных компонентов и программного обеспечения персонального компьютера; уметь применять персональные компьютеры для обработки различных видов информации;

«Английский язык» - знать лексический и грамматический минимум в объеме, необходимом для работы с текстами профессиональной направленности на иностранном языке; владеть иностранным языком в объеме, необходимом для получения и изложения информации по профессиональной тематике.

Дисциплина «Языки программирования» является предшествующей для изучения следующих базовых дисциплин: «Методы программирования», «Операционные системы», «Системы управления базами данных», «Безопасные операционные системы», «Методы и средства криптографической защиты информации». Знания и практические навыки, полученные в результате освоения дисциплины «Языки программирования», используются студентами при разработке курсовых работ, в научно-исследовательской работе.

### 3. Планируемые результаты обучения по дисциплине, соотнесенные с планируемыми результатами освоения образовательной программы

Процесс изучения дисциплины направлен на формирование следующих элементов компетенций в соответствии с ФГОС ВО, ОП ВО и приобретения следующих знаний, умений, навыков и (или) опыта деятельности:

Формируемая компетенция (код и формулировка)	Индикатор достижения компетенции (код и формулировка)	Перечень планируемых результатов обучения
<b>Общепрофессиональные компетенции</b>		
<b>ОПК-7</b> Способен использовать языки программирования и технологии разработки программных средств для решения задач профессиональной деятельности	<b>И-ОПК-7.1</b> Способен формализовать задачу, разработать алгоритм ее решения и реализовать его на языках программирования, применяя методы и инструментальные средства программирования.	<b>Знать:</b> - общие принципы построения и использования современных языков программирования высокого уровня; - базовые структуры данных; <b>Уметь:</b> - формализовать поставленную задачу; - реализовывать алгоритмы на языках программирования высокого уровня <b>Владеть:</b> - навыками документирования и сопровождения программного обеспечения с учетом повышенных требований к надежности программ и их защищенности от несанкционированного доступа
	<b>И-ОПК-7.2</b> Знает языки программирования высокого и низкого уровней и современные среды разработки программ	<b>Знать:</b> - язык программирования высокого уровня (объектно-ориентированное программирование); - язык ассемблера персонального компьютера; <b>Уметь:</b> - разрабатывать прикладное программное обеспечение для многозадачных, многопользовательских сред
	<b>И-ОПК-7.3</b> Выбирает наиболее подходящие средства и методы программирования для решения профессиональных задач	<b>Знать:</b> - особенности взаимодействия языков высокого и низкого уровня; - основные концепции языков программирования; <b>Владеть:</b> - методиками современной технологии программирования, - навыками программирования в выбранной операционной среде
	<b>И-ОПК-7.4</b> Владеет навыками кодирования, отладки, тестирования	<b>Уметь:</b> - тестировать и отлаживать программы; - работать с современными системами программирования, включая объектно-ориентированные; <b>Владеть:</b> - языками процедурного и объектно-ориентированного программирования, навыками разработки и отладки программ; - использования инструментальных средств отладки и дизассемблирования программного кода

#### 4. Объем, структура и содержание дисциплины

Общая трудоемкость дисциплины составляет **8** зачетных единиц, **288** акад. часа.

№ п/п	Темы (разделы) дисциплины, их содержание	Семестр	Виды учебных занятий, включая самостоятельную работу студентов, и их трудоемкость (в академических часах)						Формы текущего контроля успеваемости  Форма промежуточной аттестации (по семестрам)
			Контактная работа						
			лекции	практические	лабораторные	консультации	аттестационные испытания	самостоятельная работа	
1.	Вводная лекция. Основные концепции языков программирования	2	2						
2.	Формальные грамматики и языки	2	4			1		6	Контрольная работа «Эквивалентные преобразования грамматик» Контест № 11
3.	Конечные автоматы и преобразователи	2	2		4	1		10	Контрольная работа «Конечные автоматы» Индивидуальное задание на лабораторную работу № 1
4.	Автоматы и преобразователи с магазинной памятью	2	4			1		8	Контест № 12
5.	Методы синтаксического анализа	2	2			1		6	Контрольная работа «Методы синтаксического анализа»
6.	Формальные методы описания и реализации синтаксически управляемого перевода	2	4		10	1		8	Контрольная работа «ОПЗ» Индивидуальное задание на лабораторную работу № 2
7.	Общая характеристика язы- ка ассемблера	2	1					4	Контест № 13
8.	Архитектура процессора	2	1			1		2	
9.	Базовая система команд процессора	2	4		6			6	Индивидуальное задание на лабораторную работу № 3 Контест № 14
10.	Разработка программ на	2	4		6	1		6	Индивидуальное

	языке ассемблера. Средства транслятора								задание на лабораторную работу № 4 Контест № 15
11.	Режимы работы процессора. Интерфейс взаимодействия языка ассемблера и языков высокого уровня	2	4		6	1		10	Индивидуальное задание на лабораторную работу № 5
							0,3	5,7	Зачет
	<b>Всего за 2 семестр 144 акад. часа</b>		32		32	8	0,3	71,7	
12	C# и объектно-ориентированное программирование	3	6		8	1		4	Индивидуальное задание на лабораторную работу № 1, 2 Контест ООП
13	Время жизни объектов.	3	1					2	Контр. работа
14	Структурная обработка исключений	3	2		2			2	Индивидуальное задание на лаб. работу № 3
15	Интерфейсы и коллекции	3	3		4	1		4	Индивидуальное задание на лаб. работу № 4 Контест Применение различных контейнеров для данных
16	Знакомство с .Net сборками. Защита сборок. GAC.	3	2		2	1		4	Индивидуальное задание на лаб. работу № 5
17	Интерфейсы обратного вызова, делегаты и события	3	4		4	1		4	
18	Создание приложений Windows.Forms		2		2	1		4	Индивидуальное задание на лаб. работу № 6
19	Отражение типов, позднее связывание и создание расширяемых приложений	3	4		4	1		4	Индивидуальное задание на лабораторную работу № 7
20	Нетривиальные приемы создания типов в C# . Анонимные типы, инициализаторы расширяющие методы.	3	4		2	1		4	
21	Технология LINQ.	3	4		4	1		4	
						2	0,5	33,5	Экзамен
	<b>Всего за 3 семестр 144 акад. часа</b>		32		32	10	0,5	69,5	
	<b>ИТОГО</b>		64		64	18	0,8	141,2	

### Содержание разделов дисциплины:

#### Раздел 1. Основные концепции языков программирования. Язык ассемблера.

##### Тема 1. Вводная лекция. Основные концепции языков программирования.

1.1 Парадигмы ЯП. Императивные языки. Языки функционального программирования. Декларативные языки. Объектно-ориентированные языки. Критерии оценки ЯП.

- 1.2 Объекты данных в ЯП. Имена. Константы. Переменные. Механизмы типизации.
- 1.3 Время жизни переменных. Статические, динамические, явные динамические и неявные динамические переменные.
- 1.4 Область видимости переменных. Среда ссылок. Статическая область видимости. Динамическая область видимости.
- 1.5 Типы данных. Элементарные типы данных: числовые типы, логические типы, символьный тип, указатели. Символьные строки. Перечислимые типы. Ограниченные типы. Векторы и массивы. Записи. Объединения. Множества. Списки.
- 1.6 Выражения и операторы присваивания. Арифметические выражения. Логические выражения. Операторы присваивания.
- 1.7 Структуры управления на уровне операторов. Составные операторы. Условные операторы. Циклы.
- 1.8 Подпрограммы. Определение подпрограмм. Формальные и фактические параметры. Процедуры и функции. Методы передачи параметров. Сопрограммы.

## **Тема 2. Формальные грамматики и языки.**

- 2.1 Способы определения формальных языков. Алфавит. Цепочка. Язык. Способы описания языков.
- 2.2 Формальные грамматики. Классификация. Терминальные и нетерминальные символы. Правила вывода. Классификация Хомского.
- 2.3 Выводы и деревья выводов. Неоднозначность грамматик.
- 2.4 Эквивалентные преобразования КС-грамматик. Удаление бесполезных символов, устранение E-правил, исключение цепных правил, Устранение левой рекурсии.
- 2.5 Нормальная форма Хомского, Нормальная форма Грейбах.

## **Тема 3. Конечные автоматы и преобразователи.**

- 3.1 Конечный автомат. Способы задания конечных автоматов. Детерминированные конечные автоматы.
- 3.2 Автоматные грамматики. Конечные преобразователи. Решение проблем принадлежности и проблем пустоты языка для конечных автоматов. Решение проблем эквивалентности.

## **Тема 4. Автоматы и преобразователи с магазинной памятью.**

- 4.1 Расширенные МП-автоматы. Определение. Конфигурация. Эквивалентность МП-автоматов и КС-грамматик.
- 4.2 Детерминированные МП-автоматы.
- 4.3 Преобразователи с магазинной памятью.

## **Тема 5. Методы синтаксического анализа.**

- 5.1 Общие методы синтаксического анализа. Определение разбора. Нисходящий и восходящий разборы. Алгоритм Кока-Янгера-Касами для КС-грамматик. Алгоритм Эрли.
- 5.2 LL(k), LR(k) – грамматики. Алгоритмы разбора. Рекурсивный спуск. Алгоритм построения анализатора.
- 5.3 Грамматики предшествования. Понятие предшествования. Алгоритм типа «перенос-свертка». Простое, слабое и оперативное предшествование.

## **Тема 6. Формальные методы описания и реализации синтаксически управляемого перевода.**

- 6.1 Промежуточные формы представления программ. Польская запись. Тетрады. Триады. Байт-коды JVM.
- 6.2 Формальные методы описания перевода. Переводы и семантика. СУ-схемы. Транслирующие грамматики. Атрибутные транслирующие грамматики
- 6.3 Разработка и реализация синтаксически управляемого перевода.

## **Тема 7. Общая характеристика языка ассемблера**

- 7.1 Общая характеристика языков ассемблера. Назначение и применение языков ассемблера. Основные отличия языков ассемблера от языков высокого уровня. Языки ассемблера современных процессоров.

7.2 Структура языка. Элементарные конструкции. Константы. Идентификаторы. Выражения. Машинные команды и операторы языка. Основные директивы. Встроенные типы данных.

### **Тема 8. Архитектура процессора**

8.1 Микропроцессоры Intel семейства x86. Представление информации: кодирование информации, виды данных.

8.2 Структура памяти. Режимы работы процессоров. Регистры процессора. Адресация оперативной памяти. Назначение и организация стека. Вызов процедур. Обработка прерываний.

### **Тема 9. Базовая система команд процессора.**

9.1 Порядок разработки программ на языке ассемблера. Оформление программ на языке ассемблера. Базовые средства транслятора.

9.2 Формат команды, способы адресации операндов. Кодирование команд.

9.3 Команды пересылки данных.

9.4 Арифметические команды.

9.5 Логические команды. Команды сдвигов. Команды обработки битов.

9.6 Команды передачи управления.

9.7 Цепочечные команды.

### **Тема 10. Разработка программ на языке ассемблера. Средства транслятора**

10.1 Этапы разработки программ. Взаимодействие программ на языке ассемблера с операционной системой. Порядок загрузки и выполнения программ.

10.2 Отладка программ. Типовые ошибки при разработке программ на языке ассемблера.

10.3 Простые и сложные типы данных. Директивы определения новых типов данных.

10.4 Директивы описания структуры программы и ее частей. Директива задания модели памяти.

10.5 Макросредства языка ассемблера. Особенности применения макросредств. Макрорасширения и макрокоманды. Директивы условной трансляции.

### **Тема 11. Режимы работы процессора. Интерфейс взаимодействия языка ассемблера и языков высокого уровня**

11.1 Реальный режим работы процессора. Защищенный режим работы процессора. Основные отличия от реального режима. Уровни привилегированности. Организация памяти и адресация. Таблицы дескрипторов. Форматы дескрипторов. Принципы организации защиты.

11.2 Обработка прерываний в защищенном режиме. Страничная адресация. Режим виртуального процессора 8086.

11.3 Стандартные соглашения по вызову процедур. Кадр стека процедуры. Пролог и эпилог функции. Хранение аргументов и локальных переменных. Расширенный синтаксис объявления процедур. Объявление прототипов внешних процедур. Упрощенный вызов процедур.

11.4 Реализация вызовов функций языка ассемблера из программ, написанных на языках высокого уровня (ЯВУ). Реализация вызовов функций ЯВУ из программ, написанных на языке ассемблера. Взаимодействие программ на языке ассемблера с ОС.

## **Раздел 2. Объектно-ориентированное программирование**

### **Тема 12. C# и объектно-ориентированное программирование.**

12.1 Философия .NET: строительные блоки .NET (CLR, CTS и CLS). Библиотека базовых классов .NET. Преимущества C#. Типы значения и ссылочные типы. Нулевые типы.

12.2 C# и объектно-ориентированное программирование: реализация в языке инкапсуляции (понятие класса, объекта. Конструкторы. Описание своего класса и создание объектов. Роль слова static. Методы класса. Модификаторы доступа. Вызов методов у экземпляров. Свойства. Автоматические свойства.). Реализация в языке наследования. Описание класса наследника. Вызов у объекта методов базового класса и собственных. Общий предок System.Object. Запечатанные классы. Protected. Реализация в языке полиморфизма. Абстрактные классы и методы. Virtual и override. Переопределение методов базового класса. Правила приведения к базовому и производному классу

### **Тема 13. Время жизни объектов (сборка «мусора»).**

### **Тема 14. Структурная обработка исключений.**

**Тема 15. Интерфейсы и коллекции (что такое интерфейсы; стандартные интерфейсы IEnumerable, ICloneable, IComparable; обобщенные коллекции List<T>, Stack<T>, Queue<T>, Dictionary<Tkey, TValue> и т.д.)**

**Тема 16. Знакомство с .Net сборками. Защита сборок. GAC.**

**Тема 17. Интерфейсы обратного вызова, делегаты и события.**

**Тема 18. Создание приложений Windows.Forms. Событийно управляемые приложения.**

**Тема 19. Отражение типов, позднее связывание и создание расширяемых приложений.**

**Тема 20. Нетривиальные приемы создания типов в C#. Анонимные типы, инициализаторы расширяющие методы**

**Тема 21. Технология LINQ.**

## **5. Образовательные технологии, в том числе технологии электронного обучения и дистанционные образовательные технологии, используемые при осуществлении образовательного процесса по дисциплине**

В процессе обучения используются следующие образовательные технологии:

**Вводная лекция** – дает первое целостное представление о дисциплине и ориентирует студента в системе изучения данной дисциплины. Студенты знакомятся с назначением и задачами курса, его ролью и местом в системе учебных дисциплин и в системе подготовки в целом. Дается краткий обзор курса, история развития науки и практики, достижения в этой сфере, имена известных ученых, излагаются перспективные направления исследований. На этой лекции высказываются методические и организационные особенности работы в рамках данной дисциплины, а также дается анализ рекомендуемой учебно-методической литературы.

**Академическая лекция с элементами лекции-беседы** – последовательное изложение материала, осуществляемое преимущественно в виде монолога преподавателя. Элементы лекции-беседы обеспечивают контакт преподавателя с аудиторией, что позволяет привлекать внимание студентов к наиболее важным темам дисциплины, активно вовлекать их в учебный процесс, контролировать темп изложения учебного материала в зависимости от уровня его восприятия.

**Консультации** – вид учебных занятий, являющийся одной из форм контроля самостоятельной работы студентов. На консультациях по просьбе студентов рассматриваются наиболее сложные моменты при освоении материала дисциплины, преподаватель отвечает на вопросы студентов, которые возникают у них в процессе самостоятельной работы.

**Лабораторная работа** – организация учебной работы с реальными материальными и информационными объектами, экспериментальная работа с аналоговыми моделями реальных объектов.

**Контест** - соревнование по программированию, при котором участники отправляют код с решением данных задач на сайт, предназначенный для их автоматического тестирования.

## **6. Перечень лицензионного и (или) свободно распространяемого программного обеспечения, используемого при осуществлении образовательного процесса по дисциплине**

В процессе осуществления образовательного процесса по дисциплине используются:  
для формирования материалов для текущего контроля успеваемости и проведения промежуточной аттестации, для формирования методических материалов по дисциплине:

- программы Microsoft Office;
- издательская система LaTeX;



- Adobe Acrobat Reader;  
для проведения мастер-классов и подготовки материалов лекций:
- Microsoft Visual Studio 2017/2019;  
для проведения лабораторных занятий:
- MS Windows 7/10,
- Microsoft Visual Studio 2017/2019,
- FASM

## **7. Перечень современных профессиональных баз данных и информационных справочных систем, используемых при осуществлении образовательного процесса по дисциплине (при необходимости)**

- В процессе осуществления образовательного процесса по дисциплине используются:
- Автоматизированная библиотечно-информационная система «БУКИ-NEXT»  
[http://www.lib.uniyar.ac.ru/opac/bk\\_cat\\_find.php](http://www.lib.uniyar.ac.ru/opac/bk_cat_find.php)
  - Электронно-библиотечная система «Юрайт» <https://urait.ru/>
  - Электронно-библиотечная система «Университетская библиотека online»  
[www.biblioclub.ru](http://www.biblioclub.ru)
  - Электронно-библиотечная система «Лань» <http://e.lanbook.com/>

## **8. Перечень основной и дополнительной учебной литературы, ресурсов информационно-телекоммуникационной сети «Интернет» (при необходимости), рекомендуемых для освоения дисциплины**

### **а) основная литература**

1. Подбельский, В. В. Программирование. Базовый курс C# : учебник для вузов / В. В. Подбельский. — Москва : Издательство Юрайт, 2023. — 369 с. — (Высшее образование). — ISBN 978-5-534-10616-9. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: <https://urait.ru/bcode/511747>
2. Казанский, А. А. Программирование на Visual C# : учебное пособие для вузов / А. А. Казанский. — 2-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2023. — 192 с. — (Высшее образование). — ISBN 978-5-534-12338-8. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: <https://urait.ru/bcode/512404>
3. Зыков С. В. Программирование: учебник и практикум для вузов. — Москва: Издательство Юрайт, 2022. <https://urait.ru/viewer/programmirovanie-489754>
4. Якимова О. П., Якимов И. М., Дольников В. Л. Языки программирования. Лабораторный практикум. Часть 2. - Ярославль: ЯрГУ, 2012. <http://www.lib.uniyar.ac.ru/edocs/iuni/20120203.pdf>
5. М. А. Кузнецов, П. Д. Кравченя Язык ассемблер x86 учеб.-метод. пособие – Волгоград, ВолгГТУ, 2019. [https://elibrary.ru/download/elibrary\\_41597487\\_65723702.pdf](https://elibrary.ru/download/elibrary_41597487_65723702.pdf)
6. Малявко, А. А. Формальные языки и компиляторы : учебное пособие для вузов / А. А. Малявко. — Москва : Издательство Юрайт, 2022. — 429 с. — (Высшее образование). — ISBN 978-5-534-04288-7. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: <https://urait.ru/bcode/492129>
7. Толстобров А. П. Архитектура ЭВМ: учебное пособие для вузов — Москва: Издательство Юрайт, 2022. <https://urait.ru/viewer/arhitektura-evm-496167>

### **б) дополнительная литература**

1. Секаев, В. Г. Основы программирования на Ассемблере : учеб. пособие / Секаев В. Г. - Новосибирск : Изд-во НГТУ, 2010. - 100 с. - ISBN 978-5-7782-1473-6. - Текст : электронный // ЭБС "Консультант студента" : [сайт]. - URL :

- <https://www.studentlibrary.ru/book/ISBN9785778214736.html>Опалева Э. А., Самойленко В. П. Языки программирования и методы трансляции. – СПб: БВХ-Петербург, 2005.
2. Зыков, С. В. Объектно-ориентированное программирование : учебник и практикум для вузов / С. В. Зыков. — 2-е изд. — Москва : Издательство Юрайт, 2023. — 151 с. — (Высшее образование). — ISBN 978-5-534-16941-6. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: <https://urait.ru/bcode/532054>
3. Троелсен Э. Язык программирования C# 2010 и платформа .NET4 - М., Вильямс, 2011
4. Соколов В. А. Введение в теорию формальных языков: Учебное пособие. - Ярославль: ЯрГУ, 2014. <http://www.lib.uniyar.ac.ru/edocs/iuni/20140406.pdf>
5. Якимова О. П. Языки программирования. Лабораторный практикум. Часть 1. - Ярославль: ЯрГУ, 2010. <http://www.lib.uniyar.ac.ru/edocs/iuni/20100203.pdf>
6. Рихтер Д. CLR via C#: программирование на платформе Microsoft .NET Framework 4.5 на языке C# - СПб., Питер, 2017

#### **в) ресурсы сети «Интернет»**

1. Microsoft Developer Network (MSDN): <https://msdn.microsoft.com/ru-ru/>
2. НОУ «ИНТУИТ»: <http://www.intuit.ru/>
3. Конспект лекций «Методы построения компиляторов»  
[http://it.mmcs.sfedu.ru/wiki/Конспект\\_лекций](http://it.mmcs.sfedu.ru/wiki/Конспект_лекций)
4. Теория языков программирования и методы трансляции  
<http://ermak.cs.nstu.ru/trans/>
5. FASM <https://flatassembler.net/>
6. Увлекательное введение в функциональное программирование на F#  
[https://mva.microsoft.com/ru/training-courses/-f--12198?l=jkUEDSSIB\\_504984382](https://mva.microsoft.com/ru/training-courses/-f--12198?l=jkUEDSSIB_504984382)

### **9. Материально-техническая база, необходимая для осуществления образовательного процесса по дисциплине**

Материально-техническая база, необходимая для осуществления образовательного процесса по дисциплине включает в свой состав специальные помещения:

- учебные аудитории для проведения занятий лекционного типа;
- учебные аудитории для проведения лабораторных работ (лаборатория информационных систем, оборудованные ПЭВМ класса не ниже Intel Pentium II, 128 Mb RAM, 1G HDD с установленным программным обеспечением: Microsoft Windows 7, MS Office., Visual Studio 2017/2019, FASM.
- учебные аудитории для проведения групповых и индивидуальных консультаций,
- учебные аудитории для проведения текущего контроля и промежуточной аттестации;
- помещения для самостоятельной работы;
- помещения для хранения и профилактического обслуживания технических средств обучения.

Помещения для самостоятельной работы обучающихся оснащены компьютерной техникой с возможностью подключения к сети «Интернет» и обеспечением доступа к электронной информационно-образовательной среде ЯрГУ.

#### **Автор(ы):**

старший преподаватель кафедры КБ и ММОИ

Власова О. В.

доцент кафедры КБ и ММОИ, к. ф.-м. н.,

Якимова О. П.

Приложение №1 к рабочей программе дисциплины  
«Языки программирования»

**Фонд оценочных средств  
для проведения текущей и промежуточной аттестации студентов  
по дисциплине**

**1. Типовые контрольные задания или иные материалы,  
используемые в процессе текущей аттестации**

**Раздел 1.**

**Контрольная работа «Эквивалентные преобразования грамматик»**

*Пример контрольного задания*

1. Преобразуйте КС-грамматику  $G=(N, \Sigma, P, S)$  в эквивалентную грамматику, не содержащую бесполезных **символов**:

1.1.  $S \rightarrow b, S \rightarrow C, S \rightarrow cCB, A \rightarrow e, A \rightarrow Ab, B \rightarrow Bb, B \rightarrow cB, C \rightarrow Ca, C \rightarrow Bf, C \rightarrow d.$

1.2.  $S \rightarrow aC, S \rightarrow bA, A \rightarrow cAB, B \rightarrow aC, C \rightarrow bA, C \rightarrow d.$

1.3.  $S \rightarrow aABC, S \rightarrow aE, A \rightarrow SCD, A \rightarrow c, B \rightarrow bFD, C \rightarrow aE, D \rightarrow aD, B \rightarrow b, E \rightarrow aCE, E \rightarrow a, F \rightarrow AB.$

2. Преобразуйте в нормальную форму Хомского КС-грамматики  $G=(N, \Sigma, P, S)$ :

2.1.  $S \rightarrow AB, A \rightarrow SA, A \rightarrow BB, A \rightarrow bB, B \rightarrow b, B \rightarrow aA, B \rightarrow \epsilon.$

2.2.  $S \rightarrow Aa, S \rightarrow bB, A \rightarrow cAdA, A \rightarrow a, A \rightarrow \epsilon, B \rightarrow cBdd, B \rightarrow \epsilon.$

2.3.  $S \rightarrow SS, S \rightarrow 1A0, A \rightarrow 1A0, A \rightarrow \epsilon.$

3. Преобразуйте в нормальную форму Грейбах КС-грамматики  $G=(N, \Sigma, P, S)$ :

3.1.  $S \rightarrow A, S \rightarrow B, A \rightarrow 1A0, A \rightarrow 1a0, B \rightarrow 1B00, B \rightarrow 1b00.$

3.2.  $S \rightarrow Aa, S \rightarrow bB, A \rightarrow cAdA, A \rightarrow a, A \rightarrow \epsilon, B \rightarrow cBdd, B \rightarrow \epsilon.$

3.3.  $S \rightarrow abSa, S \rightarrow aaAb, S \rightarrow b, A \rightarrow baAb, A \rightarrow b.$

**Контрольная работа «Конечные автоматы»**

*Пример контрольного задания*

Построить детерминированный конечный автомат по автоматной грамматике  $G=(N, \Sigma, P, S)$ .  
Определить язык, допускаемый конечным автоматом.

4.1.  $N=\{S, A, B, C\}, \Sigma=\{a, b\}, P=\{S \rightarrow aA, S \rightarrow bB, A \rightarrow aA, A \rightarrow \epsilon, A \rightarrow bB, B \rightarrow aC, C \rightarrow aC, C \rightarrow \epsilon\}.$

4.2.  $N=\{S, A, B, C\}, \Sigma=\{a, b\}, P=\{S \rightarrow aA, S \rightarrow bS, A \rightarrow aA, A \rightarrow bB, B \rightarrow bS, B \rightarrow aC, C \rightarrow aC, C \rightarrow bC, C \rightarrow \epsilon\}.$

4.3.  $N=\{S, A, B, C, D\}, \Sigma=\{0, 1\}, P=\{S \rightarrow 1A, A \rightarrow 0S, A \rightarrow 1B, B \rightarrow 0C, C \rightarrow \epsilon, C \rightarrow 0C, C \rightarrow 1D, D \rightarrow \epsilon, D \rightarrow 0D\}.$

**Контрольная работа «Методы синтаксического анализа»**

*Пример контрольного задания*

Постройте управляющую таблицу и промоделируйте работу анализатора типа «перенос-свертка» для КС-грамматики слабого предшествования  $G=(N, \Sigma, P, S)$ , правила которой имеют вид  $S \rightarrow N; Q, N \rightarrow n, Q \rightarrow N;n, Q \rightarrow q, Q \rightarrow Q;q.$

Постройте управляющую таблицу и промоделируйте работу анализатора типа «перенос-свертка» для КС-грамматики операторного предшествования  $G=(N, \Sigma, P, S)$ , правила которой имеют вид  $S \rightarrow \text{if } E \text{ then } S \text{ else } S, S \rightarrow a, E \rightarrow E \text{ or } b, E \rightarrow b.$

Используя алгоритм Эрли, построите список разбора и разбор для КС-грамматики  $G=(N, \Sigma, P, S)$ , правила которой имеют вид  $S \rightarrow AB \mid 1, B \rightarrow 0, A \rightarrow BS \mid 1.$  Строка для анализа  $w = 0100$

**Контрольная работа «ОПЗ»**

*Пример контрольного задания*

Постройте обратную польскую запись для фрагмента программы.

По обратной польской записи построите трехадресную модель.

while (a-b)\*c <= 1 do

```

begin
  if b<c then b:=2*b-k/(1-m)
    else c:=2*b-k*(1-m);
  c:= b-a
end

```

### **Лабораторная работа №1**

**Промежуточные формы представления программ. Преобразование арифметического скобочного выражения в ПОЛИЗ. Вычисление арифметического выражения, записанного в ПОЛИЗ. Генерация промежуточного код на основе трехадресной(одноадресной). Оптимизация промежуточного кода.(C#)**

*Пример задания*

1. Представьте графически и в ПОЛИЗ выражение  
 $X := A[i, j+1] * \sin(A[i, \max(A, j+1)])$
2. Представьте графически , в обратной польской записи, в трехадресной модели оператор  

```

while (a-b)*c <= 1 do
  begin
    if b<c then b:=2*b-k/(1-m)
      else c:=2*b-k*(1-m);
    c:= b-a
  end

```
1. Запрограммировать алгоритм Дейкстры перевода арифметического скобочного выражения в ПОЛИЗ
2. Запрограммировать алгоритм вычисления арифметического выражения, записанного в ПОЛИЗ.
3. Создать транслятор выражения ПОЛИЗ в промежуточный код, представляющий собой список триад.
4. Реализовать оптимизацию промежуточного кода (свертку) за счет исключения триад, связанных с константными вычислениями и обращениями к данным

### **Лабораторная работа №2**

**Лексический анализатор на основе конечного автомата. Алгоритмы синтаксического анализа.(C#)**

*Пример задания*

1. Разработать алгоритм сканера в соответствии с заданным вариантом исходных данных  
 Ключевые слова : PROGRAM, INPUT, OUTPUT, VAR, INTEGER, BEGIN, IF, THEN, ELSE  
 Служебные знаки «:», «,», «;», «:=», «<=», «>=», «(», «)»  
 Другие лексемы: идентификаторы с количеством символов не более 5
2. Реализовать один из алгоритмов синтаксического анализа
  1. Нисходящий разбор с возвратом;
  2. Восходящий разбор с возвратом;
  3. Алгоритм Кока – Янгера - Касами;
  4. Алгоритм Эрли;
  5. Алгоритм «Перенос-свертка» для грамматики простого предшествования;
  6. Нисходящий анализ с возвратами для LL(k).

### **Лабораторная работа №3**

#### **1. Вычисление целочисленных арифметических выражений. (Ассемблер).**

**Цель работы.** Вычислить заданное целочисленное выражение для исходных данных в знаковых форматах длиной 8 и 16 бит: **ShortInt (signed char)** и **Integer (int)**, используя

арифметические операции ADD, ADC, INC, SUB, SBB, DEC, NEG, IMUL, IDIV, CBW, CWD и, если нужно, логические операции XOR, SAL, SAR.

## **2. Организация условных переходов. (Ассемблер).**

**Цель работы.** Вычислить заданное условное целочисленное выражение для данных в форматах **ShortInt (signed char)** и **Byte (unsigned char)**, используя команды сравнения, условного и безусловного перехода. Исходные данные вводятся корректно (с проверкой на область допустимых значений).

*Пример задания*

1.  $(c*b - 24 + a)/(b/2*c - 1)$ .

2. 
$$X = \begin{cases} a/b + 21, & \text{если } a > b, \\ -82, & \text{если } a = b, \\ (a-b)/a, & \text{если } a < b; \end{cases}$$

## **3. Организация циклов.**

**Цель работы.** Работа с одномерным массивом. Выполнить ввод и вывод данных с использованием подпрограмм.

1. Дан целочисленный массив из N элементов. Среди элементов массива с четными индексами, найти тот, который имеет максимальное значение.

## **4. Обработка последовательностей. Сортировка и поиск.**

**Цель работы.** Работа с одномерным динамическим массивом (выделение памяти из кучи). Выполнить ввод и вывод данных с использованием подпрограмм.

1. Задан набор натуральных чисел. Гарантируется, что все числа различны. Необходимо определить, сколько в наборе таких пар чётных чисел, что их среднее арифметическое тоже присутствует в файле, и чему равно наименьшее из средних арифметических таких пар. Входные данные представлены следующим образом. Первая строка содержит целое число N – общее количество чисел в наборе ( $N < 100$ ). Каждая из следующих N строк содержит одно число, не превышающее 104. В ответе запишите два целых числа: сначала количество пар, затем наименьшее среднее арифметическое.

*Пример входного файла:*

6  
3  
8  
14  
11  
2  
17

## **Лабораторная работа № 4**

### **Работа со строками.**

**Задание А(Ассемблер).**

Построить детерминированный конечный автомат по регулярной грамматике  $G=(N, \Sigma, P, S)$ . Определить язык, допускаемый конечным автоматом.

Написать программу реализующую работу конечного автомата по распознаванию строки (строка для анализа из файла).

**Задание В(C++, Ассемблер).**

Написать программу обработки строки используя ассемблерные вставки.

Ввод и вывод данных C++, обработка ассемблер

*Пример задания*

1.

A.

$G=(N, \Sigma, P, S)$ .  $N=\{S, A, B, C\}$ ,  $\Sigma=\{a,b\}$ ,  $P=\{S \rightarrow aA|bB|a, A \rightarrow aA|a|bB, B \rightarrow aC, C \rightarrow aC|a\}$

B.

Дана строка, состоящая из слов, набранных заглавными буквами и разделенных пробелами (одним или несколькими). Найти количество слов, которые начинаются и заканчиваются одной и той же буквой.

*Задание С(Машина Тьюринга).*

Составить программу для Машины Тьюринга.

Для сокращения формулировки задач введём следующие два соглашения:

- буквой Р будем обозначать входное слово;
- буквой А будем обозначать алфавит входного слова, т.е. набор тех символов, из которых и только которых может состоять Р (отметим, однако, что в промежуточных и выходном словах могут появляться и другие символы).

Пусть Р имеет вид Q–R, где Q и R – непустые слова из символов 0, 1 и 2.

Трактуя Q и R как записи чисел в троичной системе счисления (возможно, с незначащими нулями) и считая, что  $Q \geq R$ , выдать в качестве ответа запись разности этих чисел в той же троичной системе.

### ***Лабораторная работа № 5***

***Реализация вызовов функций языка ассемблера из программ, написанных на языках высокого уровня (C++, ассемблер)***

*Цель работы.* Использование процедур и функций в задаче на обработку двумерного массива.

Требуется вывести на экран меню, состоящее из следующих пунктов:

1. ввод матрицы с клавиатуры (C++),
2. ввод матрицы из файла(C++),
3. вычисление характеристики(Ассемблер),
4. преобразование матрицы(Ассемблер),
5. печать матрицы(C++),
6. печать матрицы в файл(C++),
7. выход.

и обеспечить его функционирование.

Внутри программы обработка всех пунктов меню оформляется в виде функций с параметрами. Тип параметров определяется исходя из смысла функции. Память под массив выделяется в функции ввода. Необходимо отслеживать, был ли произведен ввод данных до выбора пунктов меню, которые обрабатывают матрицу. Если пункт ввода матрицы выбирается повторно, то при необходимости, выполнить освобождение памяти и повторное выделение. Имя выходного файла передается через параметры функции main, а при отсутствии параметра, запрашивается у пользователя.

*Вариант характеристики.*

В матрице существует строка, элементы которой образуют симметричную последовательность.

*Вариант преобразования матрицы.*

Повернуть внешний контур матрицы на  $90^0$  по часовой стрелке, второй контур на  $90^0$  против часовой стрелки.

## **Раздел 2.**

Задания для лабораторного практикума с указанием вопросов для самопроверки приведены в учебном пособии:

Якимов О.П. Языки программирования. Лабораторный практикум. Часть 1. Ярославль: Изд.ЯрГУ, 2010. 68 с.

### ***Контрольная работа № 1. ООП.***

*Пример задания:*

#### Вариант 1.

1. В чем разница между виртуальными и абстрактными методами?
2. I. Создать класс Point(точка на плоскости с вещественными координатами ).  
II. Создать класс Circle(окружность). Поля – координаты центра и радиус.  
Конструкторы: а) через точку( центр окружности) и радиус;  
б) через две точки( одна на окружности, другая – центр).  
Методы: а) переопределить ToString(), чтобы выводилась информация об окружности ( коор-динаты центра и радиус); б) метод, определяющий касаются окружности или нет. Возвращаемое значение метода – точка касания ( объект Point). Если не касаются, то возвращается null.

#### Вариант 2.

1. Когда вызываются статические конструкторы классов в C#?
2. Создать класс Segment(отрезок на прямой). Поля – координаты концов отрезка.  
Конструкторы: а) пустой(по умолчанию); б) через координаты концов отрезка.  
Методы:  
а) переопределить ToString(), чтобы выводилась информация об отрезке;  
б) метод, определяющий пересекаются отрезки или нет. Возвращаемое значение метода – отрезок( объект Segment). Если не пересекаются, то возвращается null, иначе возвращается отрезок, являющийся пересечением.

#### ***Контрольный тест по разделу 2.***

- 1) Что понимается под термином «класс»?
- 2) Каково соотношение понятий «класс» и «объект»?
- 3) Что понимается под термином «члены класса»? Какие члены класса Вам известны? Какие члены класса содержат код, а какие данные?
- 4) Дайте определение инкапсуляции.
- 5) Каждый ли класс языка C# имеет конструктор?
- 6) С какой целью используются свойства класса?
- 7) Что понимается под термином «конструктор»? В чем состоит его назначение? Сколько конструкторов может содержать класс языка C#?
- 8) Какие модификаторы типа доступа Вам известны?
- 9) В чем заключаются особенности доступа членов класса с модификатором public? private? с модификатором protected? internal?
- 10) Зачем необходимо перегружать операции?
- 11) Что понимается под термином «наследование»?
- 12) Что общего имеет дочерний класс с родительским? В чем состоит различие между дочерним и родительским классами?
- 13) Что понимается под термином «полиморфизм»?
- 14) Какие механизмы используются в языке C# для реализации концепции полиморфизма?
- 15) С какой целью используются виртуальные методы?
- 16) В чем состоит особенность виртуальных методов в производных (дочерних) классах?
- 17) Какие условия определяют выбор версии виртуального метода?
- 18) Какое ключевое слово (модификатор) языка C# используется для определения виртуального метода в базовом (родительском) классе? А в производном (дочернем) классе?
- 19) Что понимается под термином «абстрактный класс»? В чем заключаются особенности абстрактных классов?
- 20) Являются ли абстрактные методы виртуальными?
- 21) Возможно ли создание объектов абстрактного класса?
- 22) Что такое исключение?
- 23) Какие операторы заключаются в блок try?

- 24) Сколько блоков catch может быть расположено подряд? В чем их назначение?
- 25) В чем состоит значение механизма исключений в языке C#?
- 26) Что происходит в случае неудачного перехвата исключения?
- 27) В каком случае возможно использование оператора языка C# catch без параметров?
- 28) Каким образом осуществляется возврат в программу после обработки исключительной ситуации?
- 29) Приведите синтаксис блока finally (в составе оператора try ...catch) в общем виде. Проиллюстрируйте его фрагментом программы на языке C#. Объясните применение ключевого слова finally.
- 30) С помощью какого ключевого слова происходит генерация исключений?
- 31) С какой целью создаются пользовательские исключения?
- 32) Дайте определение интерфейса. Для каких целей применяются интерфейсы?
- 33) Как можно получить ссылку на интерфейс? Приведите примеры.
- 34) В чем особенность иерархий интерфейсов?
- 35) Какие стандартные интерфейсы вы знаете? Приведите примеры их реализации.
- 36) Объясните сходства и различия классов коллекций и обобщенных коллекций. Приведите примеры обобщенных структур данных.
- 37) Чем отличается синтаксис интерфейса от синтаксиса абстрактного класса?
- 38) Какие объекты языка C# могут быть членами интерфейсов?
- 39) Может ли класс реализовывать множественные интерфейсы?
- 40) Какой модификатор доступа соответствует интерфейсу?
- 41) Возможно ли наследование интерфейсов?
- 42) Насколько синтаксис наследования интерфейсов отличается от синтаксиса наследования классов?
- 43) Что такое сборка?
- 44) В чем различие между частными и общими сборками?
- 45) Как получить сборку со строгим именем?
- 46) Каково назначение GAC ?
- 47) Что понимается под многоадресностью делегата?
- 48) В чем состоят преимущества использования делегатов?
- 49) Что является значением делегата?
- 50) В чем состоит практическое значение многоадресности?
- 51) Каким образом осуществляется создание цепочки методов для многоадресных делегатов?
- 52) Для чего служит ключевое слово event? Являются ли события членами классов?
- 53) По какой причине данный код не скомпилируется?

```
public delegate void EventHandler();
public class Control
{
    public event EventHandler Invalidate;
}
public class Button : Control
{
    public void Draw()
    {
        if (Invalidate != null)
        {
            Invalidate();
        }
    }
}
```

- 54) Как подписаться на событие и отписаться от него?



- 55) Как создать событие базового класса в производном?
- 56) Что такое отражение?
- 57) Как можно получить экземпляр класса System.Type?
- 58) Когда применяется динамическая загрузка сборок?
- 59) Для какой цели используется позднее связывание?
- 60) Какие операторы нужно переопределить у класса Digit, чтобы были возможными преобразования:

```
Digit digit = new Digit(100);  
decimal decimalDigit = digit;  
byte byteDigit = (byte)digit;
```

## **2. Список вопросов и (или) заданий для проведения промежуточной аттестации**

### **Раздел 1.**

Зачет выставляется по итогам текущей аттестации

### **Раздел 2.**

Вопросы к экзамену за III семестр по курсу «Языки программирования»

1. Философия .Net. Строительные блоки .NET (CLR, CTS и CLS) Библиотека базовых классов .NET. Преимущества C#.
2. Основы языка C#. Анатомия простой программы на C#. Варианты метода Main(). Обработка параметров командной строки. Операторы цикла, условия.
3. Определение классов и создание объектов. Определение видимости членов. Значение ключевого слова static.
4. C# и объектно-ориентированное программирование. Поддержка инкапсуляции.
5. C# и объектно-ориентированное программирование. Поддержка наследования.
6. C# и объектно-ориентированное программирование. Поддержка полиморфизма.
7. Цикл жизни объектов. Сборка мусора. Использование деструкторов.
- Интерфейс IDisposable.
8. Исключения уровня системы и исключения уровня приложения (пользовательские). Структурная обработка исключений.
9. Интерфейсы. Определение, реализация, получение ссылок на интерфейсы, явная реализация.
10. Иерархия интерфейсов. Стандартные интерфейсы IEnumerable, ICloneable, IComparable
11. Обобщенные коллекции. Их свойства, методы, применение.
12. Делегаты
13. События. Стандартный делегат для обработки событий
14. Перегрузка операторов. Преобразования типов: правила. Явные и неявные преобразования.
15. Обобщения. Создание обобщенных классов, методов, интерфейсов.
16. Знакомство с .Net сборками. Частные и общие сборки. Защита сборок.
- GAC. Настройка общих сборок.
17. Отражение типов, позднее связывание и программирование с использованием атрибутов
18. Ссылочные и значимые типы данных. System.Object. Упаковка и распаковка при преобразовании типов.
19. Ссылочные и значимые типы данных. Встроенные типы(числовые типы, char, boolean, string). Массивы. Nullable types.
20. Технология LINQ.
21. Автоматические свойства, расширяющие методы.
22. Средства языка C#: анонимные типы, инициализаторы, индексаторы.

### **3. Методические рекомендации преподавателю по процедуре оценивания знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций**

Целью процедуры оценивания является определение степени овладения студентом ожидаемыми результатами обучения (знаниями, умениями, навыками и (или) опытом деятельности).

Экзаменационный ответ оценивается по 4-х бальной системе, в соответствии с которой выставляются оценки «отлично», «хорошо», «удовлетворительно», «неудовлетворительно».

#### **Правила выставления оценки:**

оценка «отлично» выставляется студенту, если он владеет материалом дисциплины; четко и определенно отвечает на вопросы, легко сравнивает различные части, разбирает новые и сложные предлагаемые ему случаи, видит и понимает взаимосвязь понятий, легко справляется с практическими заданиями.

оценка «хорошо» выставляется студенту, если он твердо знает и понимает материал, грамотно и по существу излагает его, не допускает существенных неточностей в ответе на поставленные вопросы, правильно применяет теоретические положения при решении практических вопросов и задач, владеет необходимыми навыками и приемами их выполнения.

оценка «удовлетворительно» выставляется студенту, если он имеет знания основного материала, но не усвоил его деталей, может ответить на вопросы по определениям основных понятий, но приходит в замешательство от заданий по объяснению взаимосвязи или доказательству положений дисциплины; испытывает затруднения при выполнении практических работ.

оценка «неудовлетворительно» выставляется студенту, который не знает значительной части программного материала, допускает существенные ошибки, неуверенно, с большими затруднениями и ошибками выполняет практические работы.

#### **Оценивание контрольных работ:**

Каждая из задач(заданий) оценивается следующими баллами: 0 (задача не сделана), 1 (сделано кое-что), 2 (сделана приблизительно наполовину), 3 (сделана с некоторыми недочетами), 4 (сделана полностью). Считается общее число баллов за все задания. Оценка за работу студента ставится в зависимости от набранного им числа баллов по отношению к максимально-возможному числу баллов:

- 0 – 50 % – неудовлетворительно,
- 51 – 67 % – удовлетворительно,
- 68 – 84 % – хорошо
- 85-100% – отлично.

## Приложение №2 к рабочей программе дисциплины «Языки программирования»

### Методические указания для студентов по освоению дисциплины

Основной формой изложения учебного материала по дисциплине «Языки программирования» являются лекции, причем в форме лекции-беседы или мастер-класса. По большинству тем предусмотрены лабораторные занятия, на которых происходит закрепление лекционного материала путем применения его к конкретным задачам и отработка навыков работы по применению различных конструкций языка и структур данных. В четвертом семестре для закрепления материала необходимо выполнить проект: спроектировать и разработать приложение, игровой или обучающей направленности. При создании приложения использовать технологию WPF, приложение должно иметь сериализуемый файл настроек( или рекордов), конфигурационный файл. Проект может выполняться командой из двух человек, при этом реализуется более сложная функциональность. Защита проекта проводится на лабораторном занятии в дисплейном классе.

Для успешного освоения дисциплины очень важно решение достаточно большого количества задач, требующих разработки алгоритма и написания программы, как в аудитории, так и самостоятельно в качестве домашних заданий. Примеры решения задач разбираются на лекциях и практических занятиях, при необходимости по наиболее трудным темам проводятся дополнительные консультации. Для решения всех задач необходимо знать и понимать лекционный материал. Поэтому в процессе изучения дисциплины рекомендуется регулярное повторение пройденного лекционного материала. Материал, законспектированный на лекциях, необходимо дома еще раз прорабатывать и при необходимости дополнять информацией, полученной на консультациях, практических занятиях или из учебной литературы.

Большое внимание должно быть уделено выполнению домашней работы. В качестве заданий для самостоятельной работы дома студентам предлагаются задачи, аналогичные разобранным на лекциях и практических занятиях или немного более сложные, которые являются результатом объединения нескольких базовых задач. Задачи сдаются через систему Яндекс-контест, обязательны к решению две задачи из каждого тура.

Для проверки и контроля усвоения материала в течение обучения при сдаче лабораторных работ преподаватель задает вопросы позволяющие выяснить понимание материала. Также проводятся консультации (при необходимости) по разбору заданий для самостоятельной работы, которые вызвали затруднения.

В конце 2 семестра студенты сдают зачет, а в конце 3го и 4го - экзамен. Экзамен принимается по экзаменационным билетам, каждый из которых включает в себя один теоретический вопрос и две задачи. На самостоятельную подготовку к экзамену выделяется 3 дня, во время подготовки к экзамену предусмотрена групповая консультация.